DAM
Desarrollo de Aplicaciones Multiplataforma
2º Curso

AD
Acceso a Datos

UD 4
Modelo Objeto Relacional (ORM)

IES BALMIS
Dpto Informática
Curso 2019-2020
Versión 1 (03/2019)

UD4 – Modelo Objeto Relacional (ORM)

ÍNDICE

7. Hibernate con dos tablas

Vamos a hacer ahora un ejemplo un poco más complejo, que incluya dos tablas con una relación 1:M ("uno a muchos").

Será una evolución de la base de datos **bibliotecah**, en las que tendremos una tabla Libros y una tabla Autores. Todavía no será una solución totalmente real, porque simplificaremos suponiendo que un autor puede escribir varios libros pero que cada libro pertenece a un único autor, cosa que en muchas ocasiones no ocurre en el mundo real.

En esta ocasión, y para no mezclar con el ejercicio anterior, llamaremos a la base de datos **bibliotecah2**, y crearemos en ella una tabla **autores**, que contenga el nombre del autor y una clave primaria alfabética:

```
CREATE TABLE autores (
  cod       VARCHAR(5) PRIMARY KEY,
  nombre    VARCHAR(60)
);
```

Y una tabla "**libros**", que contendrá el título, una clave primaria numérica, y el código del autor como forma de enlazar con la otra tabla para representar esa relación "uno a muchos".

```
CREATE TABLE libros (
  id        INTEGER PRIMARY KEY,
  titulo    VARCHAR(60),
  codautor  VARCHAR(5),
  FOREIGN KEY (codautor) REFERENCES autores(cod)
);
```

Y añadimos un par de datos de ejemplo en cada tabla:

```
INSERT INTO autores (cod, nombre) VALUES
  ('WSHAK', 'William Shakespeare'),
  ('MCERV', 'Miguel de Cervantes'),
  ('FROJA', 'Fernando de Rojas');

INSERT INTO libros (id, titulo, codautor) VALUES
  (1, 'Macbeth', 'WSHAK'),
  (2, 'La Celestina (Tragicomedia de Calisto y Melibea)', 'FROJA'),
  (3, 'Don Quijote de la Mancha', 'MCERV'),
  (4, 'La tempestad', 'WSHAK'),
  (5, 'La Galatea', 'MCERV'),
  (6, 'Los trabajos de Persiles y Sigismunda', 'MCERV'),
  (7, 'Novelas ejemplares', 'MCERV');
```

Podemos comprobar que los datos son correctos con

```
SELECT *
  FROM autores, libros
  WHERE autores.cod = libros.codautor;
```

Los pasos para crear el proyecto de Hibernate serán básicamente los mismos que el proyecto de una sola tabla:

- Comenzaremos un proyecto vacío de tipo "Java Application", con el nombre Hibernate2, con lo que se generará una clase vacía.
- Después crearemos el fichero de configuración de Hibernate, de la misma forma que lo hicimos para el proyecto de una sola tabla: indicando que deseamos un “nuevo fichero ("New File"), entrando a la categoría Hibernate y seleccionando el “Hibernate Configuration Wizard”. Crearemos una nueva conexión, de tipo MySQL, a nuestra nueva base de datos BibliotecaH2, con el usuario y contraseña que hayamos escogido.
- A continuación crearemos el fichero de ingeniería inversa, con el "Hibernate Reverse Engineering Wizard", añadiendo las dos tablas que habíamos creado en MySQL

Ahora usaremos el asistente "Hibernate Mapping Files and POJOs From Database" para crear las clases y el fichero de mapeado directo. Detectará el fichero de configuración de Hibernate y el de ingeniería inversa utilizar pero nos obligará a indicar un nombre de "package", que podría ser "hibernate2" (similar al del proyecto, pero en minúsculas). Se crearán dos ficheros para las dos nuevas clases, "Autores.java" y "Libros.java", así como los dos ficheros de mapeado directo "Autores.hbm.xml" y "Libros.hbm.xml".

La clase "Libros" será muy similar a la de antes, excepto por el detalle de que el autor ya no será un "String", sino un dato de tipo "Autor":

```
package hibernate2;

public class Libros java.io.Serializable {
  private int id;
  private Autores autores;
  private String titulo;

  public Libros() { }

  public Libros(int id) {
    this.id = id;
  }

  public Libros(int id, Autores autores, String titulo) {
    this.id = id;
    this.autores = autores;
    this.titulo = titulo;
  }
```

```
  public int getId() {
    return this.id;
  }
  public void setId(int id) {
    this.id = id;
  }

  public Autores getAutores() {
    return this.autores;
  }
  public void setAutores(Autores autores) {
    this.autores = autores;
  }

  public String getTitulo() {
    return this.titulo;
  }
  public void setTitulo(String titulo) {
    this.titulo = titulo;
  }
}
```

En la clase "Autores", dado que cada autor puede escribir más de un libro, su lista de libros será un conjunto (un "Set"):

```
package hibernate2;

import java.util.HashSet;
import java.util.Set;

/** * Autores generated by hbm2java */
public class Autores implements java.io.Serializable {
  private String cod;
  private String nombre;
  private Set libroses = new HashSet(0);

  public Autores() { }

  public Autores(String cod) {
    this.cod = cod;
  }

  public Autores(String cod, String nombre, Set libroses) {
    this.cod = cod;
    this.nombre = nombre;
    this.libroses = libroses;
  }

  public String getCod() {
    return this.cod;
  }
  public void setCod(String cod) {
    this.cod = cod;
  }
```

```
  public String getNombre() {
    return this.nombre;
  }
  public void setNombre(String nombre) {
    this.nombre = nombre;
  }

  public Set getLibroses() {
    return this.libroses;
  }
  public void setLibroses(Set libroses) {
    this.libroses = libroses;
  }
}
```

Para poder añadir la lógica del proyecto, creamos el correspondiente "**Biblio2HibernateUtil**" con el asistente correspondiente. A partir de entonces ya podemos crear un primer proyecto que muestre los datos de un libro:

```
package hibernate2;

import org.hibernate.Session;
import org.hibernate.Query;
import java.util.List;

public class Hibernate2 {
  private static Session sesion;

  // verLibros
  public static void verLibros() {
    System.out.println("Mostrando todos los libros:");
    System.out.println("---------------------------");

    Query consulta = sesion.createQuery("from Libros");

    List resultados = consulta.list();
    for(Object resultado : resultados) {
      Libros libro = (Libros) resultado;
      System.out.println ( libro.getId() + ": " + libro.getTitulo()
                + ", de " + libro.getAutores().getNombre());
    }

  }

  // main
  public static void main(String[] args) {
    org.jboss.logging.Logger logger =
                org.jboss.logging.Logger.getLogger("org.hibernate");
    java.util.logging.Logger.getLogger("org.hibernate")
                .setLevel(java.util.logging.Level.SEVERE);

    sesion = Biblio2HibernateUtil.getSessionFactory().openSession();

    verLibros();

    sesion.close();
    Biblio2HibernateUtil.closeSessionFactory();
  }

}
```

La única diferencia con el método "**verLibros()**" del anterior proyecto **Hibernate1** es que ahora no se usa "**getAutor()**" para obtener el nombre del autor, sino "**getAutores().getNombre()**".

Si los datos que obtenemos son los de los autores, recibiremos una colección de libros para cada autor, que deberemos recorrer, como en este ejemplo:

```
public static void verAutores() {

  System.out.println("Mostrando todos los autores y sus libros:");
  System.out.println("-----------------------------------------");
  Query consulta = sesion.createQuery("from Autores");

  List resultados = consulta.list();
  for(Object resultado : resultados) {
    Autores autor = (Autores) resultado;
    System.out.println(autor.getNombre()+" ("+autor.getCod()+ "). Libros: ");
    for(Object l : autor.getLibroses())
      System.out.println(" => " + ((Libros) l).getTitulo());

    System.out.println();
  }

}
```